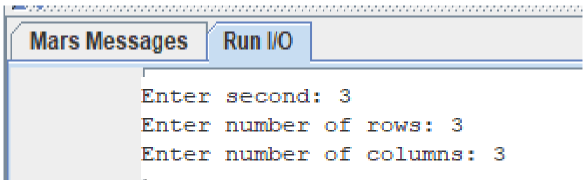
CSE 331 – Computer Organization – HW1 Report
Alper Tavşanoğlu – 210104004142

First of all, I am taking the second, row and column values from the user.

```
# Print for second
li $v0, 4                   # syscall: print_str
la $a0, prnt3               # load address of prnt2
syscall                     #system call for printing string

# Read second
li $v0, 5                   # syscall: read_int
syscall                     #system call for reading integer
move $s0,$v0                # s0 = sec
```



```
Enter second: 3
Enter number of rows: 3
Enter number of columns: 3
```

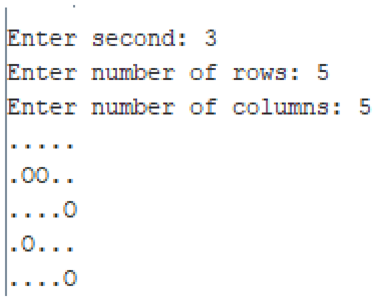
(User pressing enter after entered second , row and columns)

(This code part just example for reading second)

I defined a 2D array **using sbrk** (for dynamic memory allocation) according to these row and column values received from the user. Syscall number for sbrk is 9. And $to holds base address of 2D array.

```
mul $a0, $t1, $t2           # row * columns
li  $v0, 9                  # allocate heap memory for 2D array
syscall                     #system call for sbrk (allocate heap memory)
move $t0,$v0                # save array base address in $t0
```

After defined 2D array, I am reading user input with nested loop (inner and outer) for fill 2D array.



```
Enter second: 3
Enter number of rows: 5
Enter number of columns: 5
.....
.OO..
....O
.O...
....O
```

I am using " . " (Ascii 46) and " O " (capital O)(Ascii 79) for inputs.

The user can enter as many inputs as the number of column in each row. This ensures that the input is visually separated into rows.

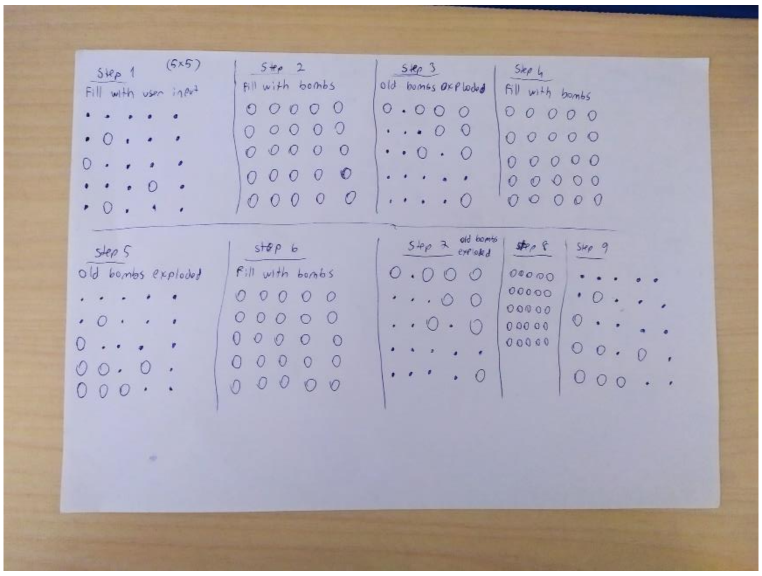
(There is no need press enter or anything after rows. Just enter " . " or " O " . Program automatically moves to the new line section.)

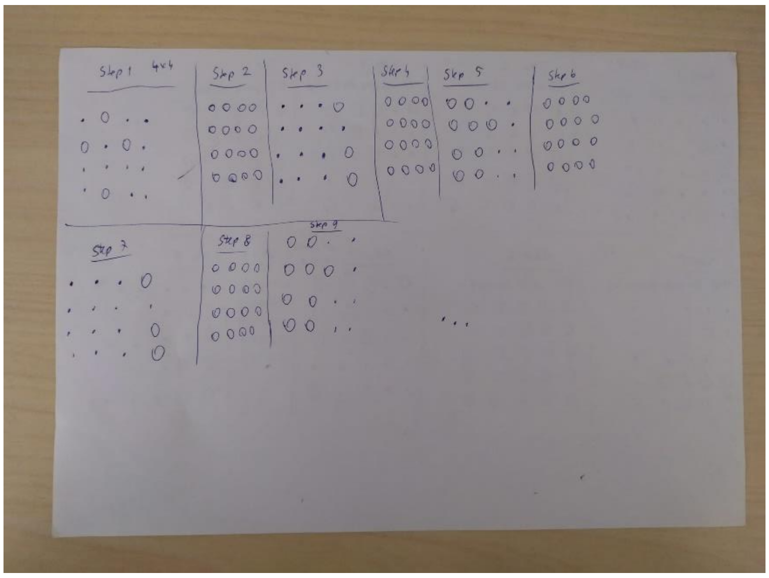
Then, I am cheking second. If **second is 1**, we need to print initial grid which user entered. Therefore program jump to **printing finale array** subroutine.

If **second is even** (i am cheking with mod 2), the whole grid must be bombed. Therefore program jump to **enter bomb all grid** subroutine for even seconds. Then program jump to **printing finale array** subroutine.

```
############################################## for cheking second conditions
    li $t3,1                                   # load t3 = 1 for checking second
    beq $s0,$t3,print_finale_array             # if second = 1 print starting grid which user entered with subroutine

    li $t3,2                                   # load t3 = 2 for checking second
    div $s0,$t3                                # divide for checking second mod 2
    mfhi $t3                                   # store remainder
    beq $t3,$zero,enter_bomb_all_grid          # if second % 2 == 0  enter bomb for all grid subroutine
```

If **second is odd** , we have to check (second+1)%4==0 condition (this is because, in this algorithm the grids for *N=3* and *N=7* are equal also N=5 and N=9 are equal). So if (second+1)%4==0 $s2 holds 1 otherwise $s2 holds 2 for **jump and running $s2 times bomberman game subroutine.I** (1 time or 2 time).





In bomberman subroutine part, I am cheking current cell contain bomb or not with nested loop.

If current cell has no bomb, it pass to next cell and check all grid.

If there is a bomb in cell, neighboring cells must explode. I am reaching right, left, top and under cells and changing them (if not alread “**.**” Symbol) with “ **/** “ symbol .

This part, i used so much in program, It allows to access the elements of the 2D array

```
# Calculate the index in the matrix
mul $t8, $t3, $t2                       # $t8 = width * i
add $t8, $t8, $s4                       # $t8 = width * i + j
sll $t8, $t8, 2                         # $t8 = 2^2 * (width * i + j)
add $t8, $t0, $t8                       # $t8 = base address + (2^2 * (width * i + j))
```

For example, here i am checking the neighboring cell to the **right** of the cell containing the bomb.

```
addiu $s4, $t4, 1                       # j + 1 for reach right side
slt  $s5, $s4,$t2                       # check  j + 1 < col
beq $s5, $zero, Resume2                 # if not  j + 1 < col  jump condition 2
# Calculate the index in the matrix
mul $t8, $t3, $t2                       # $t8 = width * i
add $t8, $t8, $s4                       # $t8 = width * i + j
sll $t8, $t8, 2                         # $t8 = 2^2 * (width * i + j)
add $t8, $t0, $t8                       # $t8 = base address + (2^2 * (width * i + j))
lb $t9, 0($t8)                          # store grid[i][j+1]
bne $t9, $t7, Resume2                   # if grid[i][j+1] !=46 ( which . )  jump condition 2
li $s6,47                               # make grid[i][j+1] = temporary symbol  "/"
sb $s6, 0($t8)                          # Store the character in memory
```

I am cheking right and left like this, for top and under i am using height instead of width.

After changed neighboring cells, if cells contains **“.”** i am changing them with **“O”** . If cells contains **other than “.”** (so exploded) i am changing them with **“.”** .

```
    # Calculate the index in the matrix
    mul $t5, $t3, $t2                    # $t5 = width * i
    add $t5, $t5, $t4                    # $t5 = width * i + j
    sll $t5, $t5, 2                      # $t5 = 2^2 * (width * i + j)
    add $t5, $t0, $t5                    # $t5 = base address + (2^2 * (width * i + j))
    lb $t8, 0($t5)                       # t8 = initial grid
    bne $t8, $t7, change                 # if initial grid !=46 ( which . )  jump condition change
    sb $t6, 0($t5)                       # if initial grid ==46 ( which . )  make it 79 ( which O )
    j skip                               # jump skip
change:
    sb $t7, 0($t5)                       #  if initial grid !=46 ( which . ) make it  46 ( which . )
skip:
    addiu $t4, $t4, 1                    #  increment inner loop counter
    b ic_loop_degisim                    # branch unconditionally back to the beginning of the inner loop
```

$t6 contains (Ascii 79 “O”)

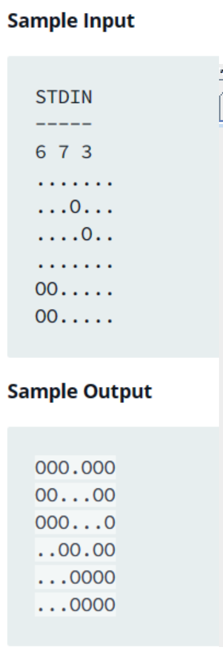
$t7 contains (Ascii 46 “.”)

After the changing part program jump again **bomberman subroutine** part or **printing finale array subroutine** according to **$s2** (i explained odd second part).

And in **printing finale array subroutine** part, i am printing whole array according to user second.

Examples:

Example From Hackerrank

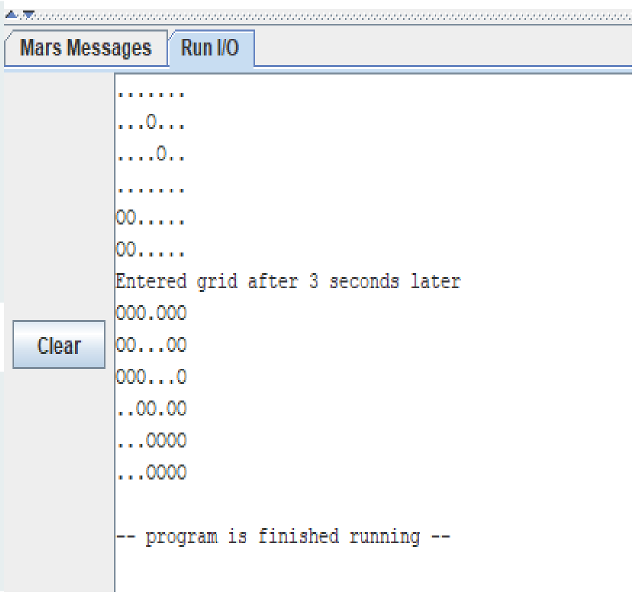


Sample Input

```
STDIN
-----
6 7 3
.......
...O...
....O..
.......
OO.....
OO.....
```

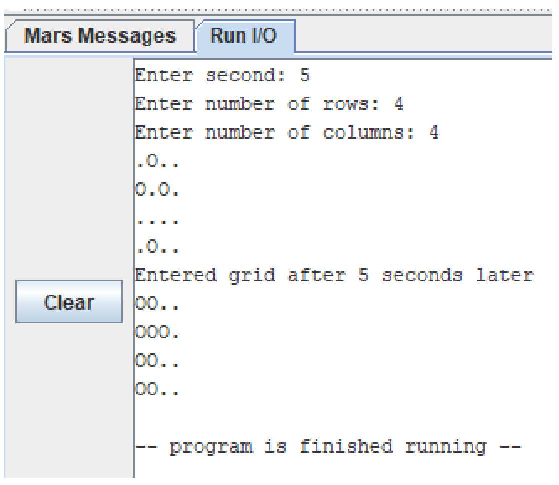
Sample Output

```
OOO.OOO
OO...OO
OOO...O
..OO.OO
...OOOO
...OOOO
```

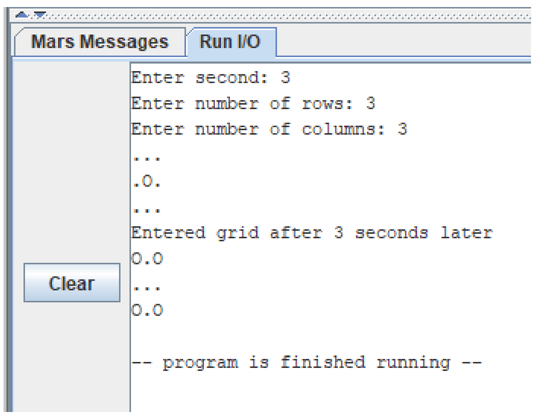


```
.......
...O...
....O..
.......
OO.....
OO.....
Entered grid after 3 seconds later
OOO.OOO
OO...OO
OOO...O
..OO.OO
...OOOO
...OOOO

-- program is finished running --
```

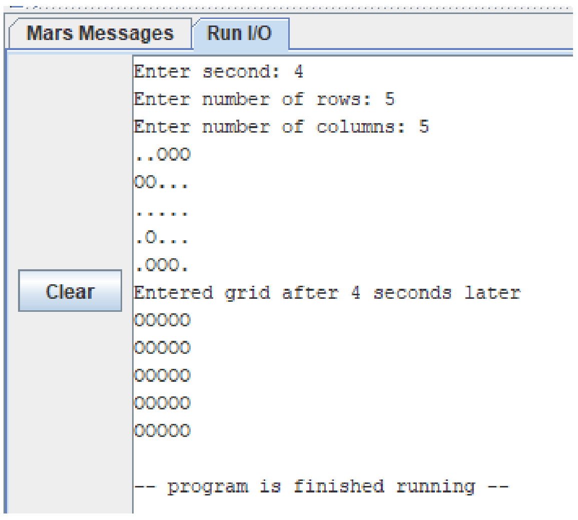
Other Examples



```
Enter second: 5
Enter number of rows: 4
Enter number of columns: 4
.O..
O.O.
....
.O..
Entered grid after 5 seconds later
OO..
OOO.
OO..
OO..

-- program is finished running --
```



```
Enter second: 3
Enter number of rows: 3
Enter number of columns: 3
...
.O.
...
Entered grid after 3 seconds later
O.O
...
O.O

-- program is finished running --
```



```
Enter second: 4
Enter number of rows: 5
Enter number of columns: 5
..OOO
OO...
.....
.O...
.OOO.
Entered grid after 4 seconds later
OOOOO
OOOOO
OOOOO
OOOOO
OOOOO

-- program is finished running --
```